COMPUTATIONAL MORPHOLOGY

(The two-level paradigm)
Sample FSA’s

- Spanish adjectives
- Uighur verbs
- French verbs
- Albanian declension
- Daily activities
- Traffic lights in Romania
- Beatles song lyrics
- Arabic perfective 3rd person verbs
- How to make friends
Finite-state morphology engines

- PC-Kimmo: original, time-tested
- xfst: industrial-strength, need license
- foma: flexible, intuitive, Unicode, regex, gviz
- OpenFST: Google, NYU
- Kleene: bleeding-edge
Two-level rules

- Serve as filters during analysis or generation
- Format: change arrow environment
- Are compiled up into FS tables, FSA
  - Limited regular expressions can be used
- Processing steps through FSA when considering each letter
  - Failure: automaton doesn’t reach end state
  - Success: processing can continue to other letters
- Operate in parallel
- Various types of rules: <=, =>, <=>, /<=
PC-KIMMO
The right-arrow rule (obna)

L:S => E  "Only but not always."
  L is realized as S only in E.
  L realized as S is not allowed in ¬E.
If L:S, then it must be in E.
Implies L:¬S in E is permitted.
The left-arrow rule (abno)

L:S \leq E \quad "Always but not only."
L is always realized as S in E.
L realized as \neg S is not allowed in E.
If L is in E, then it must be L:S.
Implies L:S may occur elsewhere.
The double-arrow rule

L:S <=> E  "Always and only."
L is realized as S only and always in E.
Both L:S => E and L:S <= E.
Implies L:S is obligatory in E and occurs nowhere else.
The never rule

L:S /<= E  "Never."
L is never realized as S in E.
L realized as S is not allowed in E.
If L is in E, then it must be L:¬S.
Specifying the environment

• Just like in phonology/morphology:

\[ X \rightarrow Y / LHContext \quad __ \quad RHContext \]

• e.g.

\[ [b] \rightarrow [p] / V \quad __ \quad # \]
The rule file

- Declare the alphabet (characters to be used in the word forms)
- Declare special symbols (null, wildcard, word boundary)
- Declare special subsets
- Declare a default rule
Declarations & default rules

ALPHABET
  p t k b d g m n ng s l r w y h ? R X N +
  a i u E
NULL 0
ANY @
BOUNDARY #
SUBSET C p t k b d g m n ng s l r w y R ; consonants
SUBSET V a i u E ; vowels
SUBSET P p b ; labial stops
SUBSET T t d ; dental stops
SUBSET K k g ; velar stops
SUBSET NAS m n ng ; nasals

RULE "1 Consonant Defaults" 1 18
  p t k b d g m n ng s l r w y h ? + @
  p t k b d g m n ng s l r w y h ? 0 @
1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

RULE "2 Vowel Defaults" 1 4
  a i u @
  a i u @
1: 1 1 1 1
Rules and tables

LR:  ap+ma  ap+ma  ap+ba
SR:  ab0ma  ap0ma  ap0ba

R35  p:b => ___ +:0 m

p + m @
b 0 m @
-------
1: 2 1 1 1
2. 0 3 0 0
3. 0 0 1 0
Russian rule

RULE "1:0 <=> @:CN (CN:0) +:0 ___ #" 4 7
    @  CN  l  l  #  +  @
    CN  0  0  @  #  0  @
1: 2  1  0  1  1  1  1
2: 2  2  4  3  1  2  1
3: 2  1  4  1  0  3  1
4: 0  0  0  0  1  4  0

; nes+l  rost+l
; nes00  ros000  (l-drop)
Two Japanese rules (simplif.)

; Voicing, t:d
; LR: Sin+ta  yom+ta  yob+ta
; SR: Sin0da  yon0da  yon0da

RULE t:d <=> M:@ (+:0) _

; Nasalization, B:n
; LR: yom+ta  yob+ta
; SR: yon0da  yon0da

; @:d due to t:d rule
RULE B:n <=> _ (+:0) @:d
Japanese subsets (classes)

SUBSET C p t k b d g m n s z j h r w y S T &
SUBSET V i e a o u
SUBSET M b g m n
SUBSET B b m
SUBSET K k g
SUBSET X s k g
SUBSET A a i
SUBSET I i u o
SUBSET R r w
An English rule (simplified)

; Epenthesis
; LR: fox+0s  kiss+0s  church+0s  spy+0s  hero+0s
; SR: fox0es  kiss0es  church0es  spi0es  hero0es

RULE 0:e <= [C|sh|y:i] +:0 _ s           (abno)
RULE 0:e => [C|ch|sh|y:i|o] +:0 _ s         (obna)
Compiling rules

• Compiling a rule consists of creating its FS table
• Wasn’t computationally feasible until late 1980’s (Xerox PARC)
• Rules were compiled by hand
• Limited capability supplied by Kgen
Lexicons

- Lexicons are used for affixes, roots
- Lexical analysis of a surface form is a path of lexical letters through one or more lexicons
- Any analysis not supported by the lexicons fails
Lexicon architecture

- Sequence of morpheme lexicons to control search through word positions
- Paths of lexical letters licensed by lexicons

Diagram:

- Vpref
- DetPref
- Nomz
- RedupPref
- Root
- RedupPost
- LxSuff
- VsufTrx
- VsufAsp
- DetSuf
Word grammar

- Specify unification-based constraints on morphemic cooccurrence
- Impose structure on parses
- Provide and display glosses
- Control percolation of featural information
- About 30 per language, quite dissimilar
Sample rule, table, FSA

;;; Optional syncope rule
;;; Note: free variation
;;; L: Lu+ad+s+pastEd
;;; S: L00ad0s0pastEd

RULE
"u:0 => [L|T'] ___ +:@ VW" 4 6
Sample word-formation rules

NWord -> { VFrame / VWord / NWord / RootX } DET2

Word -> VWord
    <Word tense> = <VWord tense>
    <Word aspect> = <VWord aspect>
    <Word inflected> = <VWord inflected>
    <Word head> = <VWord head>
foma

8.7 MB. 105926 states, 567208 arcs, more than 9223372036854775807 paths.

(1)   try +V +3P +Sg   try +V +PastPart (lex in)
(2)   try ^ s         try ^ ed       (lex out)
       trie ^ s         trie ^ ed      (desired rule output)
Sample foma English rules

# E deletion: silent e dropped before -ing and -ed (make/making)
define EDeletion e -> 0 || _ "^" [ i n g | e d ] ;

# E insertion e added after -s, -z, -x, -ch, -sh before s (watch/watches)
define EInsertion [ .. ] -> e || s | z | x | c h | s h _ "^" s ;

# Y replacement: -y changes to -ie before -s, -i before -ed (try/tries)
define YReplacement y -> i e || _ "^" s , ,
    y -> i || _ "^" e d ;

# K insertion: verbs ending with vowel + -c add -k (panic/panicked)
define KInsertion [ .. ] -> k || V c _ "^" [ e d | i n g ] ;

# Cleanup: remove morpheme boundaries
define Cleanup "^" -> 0 ;
Related concepts

- Tape analogy
- Cascade of transducers: intersecting, composing
- OCR
- Edit distance, alignment
- Dynamic programming
- Human morphological processing
- Bakeoff